ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

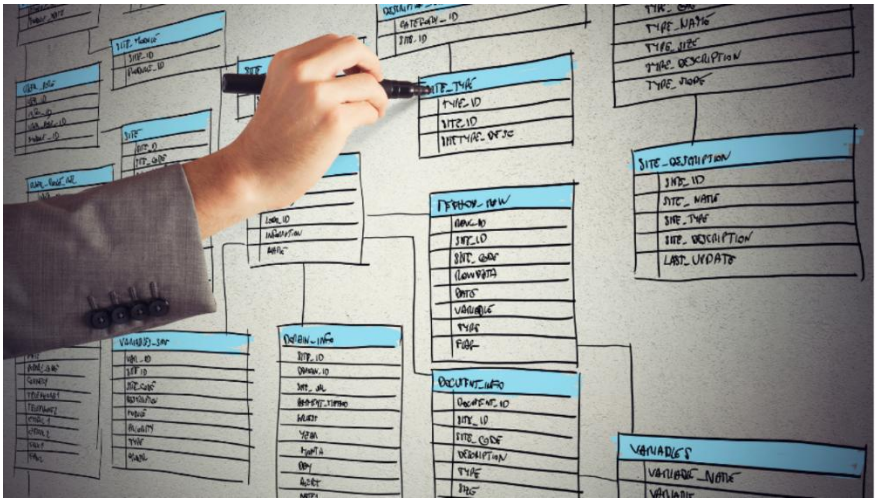
BASE DE DATOS

PROFESOR:	Ing. Yadira Franco R
PERÍODO ACADÉMICO:	2024-B

TAREA

TÍTULO:

INVESTIGACIÓN Y PRACTICA



Estudiante

Christian Márquez

2024-B

INVESTIGAR QUE SON Procedimientos Almacenados en Bases de Datos

- Entender qué son los procedimientos almacenados y cómo funcionan.
- Aprender a crear procedimientos almacenados sencillos.
- PRACTICA - Realizar operaciones de **INSERT**, **SELECT**, **DELETE** y **UPDATE** usando procedimientos almacenados.
- **Revisión de Buenas Prácticas**

Introducción a los Procedimientos Almacenados MSQL- PostgreSQL – Sql Server

1. Concepto y Beneficios de los Procedimientos Almacenados

- **Explicación**: Los procedimientos almacenados son conjuntos de instrucciones SQL que se guardan y ejecutan en el servidor de base de datos. Permiten ejecutar operaciones complejas, con seguridad, rendimiento optimizado y reutilización de código.
- **Beneficios**:

  Reutilización de código.

  Mejora en la seguridad (al evitar inyecciones SQL).

  Optimización en el rendimiento de consultas frecuentes.

  Consistencia en las operaciones realizadas.

2. ESPECIFICAR LA Sintaxis Básica de un Procedimiento Almacenado

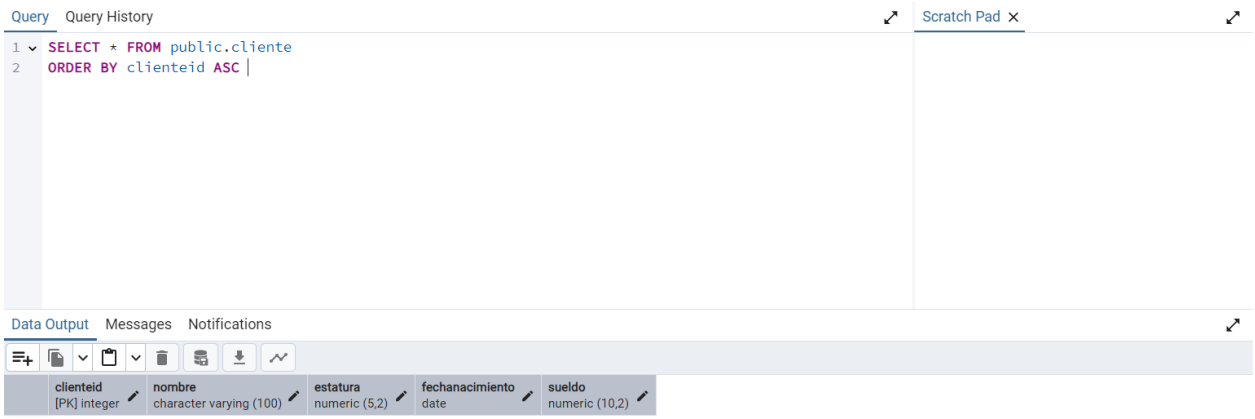
- **Explicación**: El delimitador se cambia temporalmente para permitir el uso de ; dentro del procedimiento.

Crear la tabla de cliente:

```
CREATE TABLE cliente (
    ClienteID SERIAL PRIMARY KEY,
    Nombre VARCHAR(100),
    Estatura DECIMAL(5,2),
    FechaNacimiento DATE,
    Sueldo DECIMAL(10,2)
```

- ClienteID: Se utiliza SERIAL para la columna ClienteID en lugar de AUTO_INCREMENT, ya que es el tipo de datos en PostgreSQL para generar valores automáticos secuenciales.
- Nombre: Es un campo de tipo VARCHAR(100) para almacenar el nombre del cliente, con una longitud máxima de 100 caracteres.

- Estatura: Es un campo de tipo DECIMAL(5,2), lo que indica que se pueden almacenar hasta 5 dígitos con 2 decimales. Esto es adecuado para la estatura de una persona.
- FechaNacimiento: Se usa el tipo DATE para almacenar la fecha de nacimiento del cliente.
- Sueldo: Similar a Estatura, el campo Sueldo es de tipo DECIMAL(10,2), permitiendo almacenar sueldos con hasta 10 dígitos, de los cuales 2 son decimales.



3. Ejercicio 1: Crear un procedimiento simple que seleccione datos de la tabla cliente

```
9  --2. Ejercicio 1: Procedimiento para seleccionar datos de la tabla cliente
10 CREATE OR REPLACE FUNCTION seleccionarClientes()
11 RETURNS TABLE(ClienteID INT, Nombre VARCHAR, Estatura DECIMAL, FechaNacimiento DATE, Sueldo DECIMAL) AS
12 $$
13 BEGIN
14     RETURN QUERY
15     SELECT c.ClienteID, c.Nombre, c.Estatura, c.FechaNacimiento, c.Sueldo
16     FROM cliente c;  -- Alias "c" para la tabla cliente
17 END;
18 $$ LANGUAGE plpgsql;
19
20
21 --3. Ejercicio 2: Ejecutar el procedimiento
22 SELECT * FROM seleccionarClientes();
23
```

- CREATE OR REPLACE FUNCTION: Esto define o reemplaza una función existente en PostgreSQL.
- RETURNS TABLE(...): Aquí definimos que la función devolverá una tabla con los campos ClienteID, Nombre, Estatura, FechaNacimiento, y Sueldo, que son los datos que vamos a seleccionar de la tabla cliente.
- RETURN QUERY: La función ejecuta la consulta SQL dentro de RETURN QUERY para obtener los registros de la tabla cliente.
- FROM cliente c: Usamos un alias c para la tabla cliente para evitar ambigüedades con los parámetros o posibles nombres de variables dentro del procedimiento.

- LANGUAGE plpgsql: Especificamos que la función está escrita en el lenguaje PL/pgSQL de PostgreSQL, que es un lenguaje de procedimientos para crear funciones almacenadas.

4. **Ejercicio**: Ejecutar - LLAMAR el procedimiento

Inserción, Actualización y Eliminación de Datos

1. Procedimiento de Inserción (INSERT)

- Crear un procedimiento que permita insertar un nuevo cliente en la tabla cliente

- Ejecutar - LLAMAR el procedimiento

Query | Query History

```
24  --4. Procedimiento de Inserción (INSERT)
25  CREATE OR REPLACE FUNCTION insertarCliente(
26      p_nombre VARCHAR(100),
27      p_estatura DECIMAL(5,2),
28      p_fechaNacimiento DATE,
29      p_sueldo DECIMAL(10,2)
30  )
31  RETURNS VOID AS
32  $$
33  BEGIN
34      INSERT INTO cliente (Nombre, Estatura, FechaNacimiento, Sueldo)
35      VALUES (p_nombre, p_estatura, p_fechaNacimiento, p_sueldo);
36  END;
37  $$ LANGUAGE plpgsql;
38
39  SELECT insertarCliente('Juan Pérez', 1.75, '1985-08-10', 30000.00);
```

Data Output | Messages | Notifications

Showing rows: 1 to 1 | Page No:

	insertarcliente void
1	

- Parámetros: La función insertarCliente recibe 4 parámetros (p_nombre, p_estatura, p_fechaNacimiento, p_sueldo) que contienen los valores a insertar en la tabla cliente.
- INSERT INTO: Usamos la instrucción SQL INSERT INTO para agregar un nuevo registro a la tabla cliente, usando los valores de los parámetros de la función.
- RETURNS VOID: Esto indica que la función no devuelve ningún valor. Solo realiza la operación de inserción.

2. Procedimiento de Actualización (UPDATE)

Actualizar la edad de un cliente específico:

```sql
--5. Procedimiento de Actualización (UPDATE)
CREATE OR REPLACE FUNCTION actualizarEdadCliente(
    p_clienteID INT,
    p_nuevaEdad INT
)
RETURNS VOID AS
$$
BEGIN
    UPDATE cliente
    SET Edad = p_nuevaEdad
    WHERE ClienteID = p_clienteID;
END;
$$ LANGUAGE plpgsql;
```

Data Output | Messages | Notifications

```
CREATE FUNCTION

Query returned successfully in 113 msec.
```

- UPDATE: Actualiza la columna Edad de la tabla cliente, según el ClienteID que se pasa como parámetro.
- RETURNS VOID: La función no devuelve valores, solo realiza la operación de actualización.

3. Procedimiento de Eliminación (DELETE)

Eliminar un cliente de la base de datos usando su ClienteID:

```sql
--6. Procedimiento de Eliminación (DELETE)
CREATE OR REPLACE FUNCTION eliminarCliente(
    p_clienteID INT
)
RETURNS VOID AS
$$
BEGIN
    DELETE FROM cliente WHERE ClienteID = p_clienteID;
END;
$$ LANGUAGE plpgsql;
SELECT eliminarCliente(5);  -- Suponiendo que 5 es el ID del cliente a eliminar
```

Data Output | Messages | Notifications

Showing rows: 1 to 1 Page No:

	eliminarcliente void
1	

- DELETE FROM: Elimina un registro de la tabla cliente donde el ClienteID coincide con el valor del parámetro p_clienteID.
- RETURNS VOID: La función no devuelve ningún valor.

Introducción a Condiciones en Procedimientos Almacenados

Uso de Condicionales (IF)

El uso de condicionales dentro de los procedimientos es fundamental para tomar decisiones basadas en los datos.

Verifica si la edad de un cliente es mayor o igual a 22:

Query Query History

```
--7. Uso de Condicionales (IF) en Procedimientos Almacenados
CREATE OR REPLACE FUNCTION verificarEdadCliente(
    p_clienteID INT
)
RETURNS VOID AS
$$
DECLARE
    clienteEdad INT;
BEGIN
    SELECT EXTRACT(YEAR FROM age(FechaNacimiento)) INTO clienteEdad
    FROM cliente WHERE ClienteID = p_clienteID;

    IF clienteEdad >= 22 THEN
        RAISE NOTICE 'El cliente es mayor de 22 años';
    ELSE
        RAISE NOTICE 'El cliente es menor de 22 años';
    END IF;
END;
$$ LANGUAGE plpgsql;

SELECT verificarEdadCliente(3);
```

Data Output Messages Notifications

Showing rows: 1 to 1 Page No: 1

	verificaredadcliente void
1	

- DECLARE: Declaramos una variable local llamada clienteEdad para almacenar la edad del cliente.
- SELECT EXTRACT(YEAR FROM age(FechaNacimiento)) INTO clienteEdad: Calculamos la edad del cliente en años a partir de su fecha de nacimiento.
- IF: Usamos un condicional para verificar si la edad del cliente es mayor o igual a 22 años, y luego mostramos un mensaje con RAISE NOTICE.

Creación de la Tabla de Órdenes CON RELACIÓN CON EL CLIENTE - FORANEA

Para almacenar las órdenes de los clientes, se debe crear la tabla **ordenes**:

```
--8. Creación de la Tabla de Órdenes (con relación con el Cliente)
CREATE TABLE ordenes (
    OrdenID SERIAL PRIMARY KEY,
    ClienteID INT,
    FechaOrden DATE,
    Monto DECIMAL(10,2),
    FOREIGN KEY (ClienteID) REFERENCES cliente(ClienteID)
);
```

Data Output Messages Notifications

```
CREATE TABLE

Query returned successfully in 119 msec.
```

- **OrdenID SERIAL**: Usamos SERIAL para la clave primaria OrdenID, lo que genera un valor automático único.
- **ClienteID INT**: Es una clave foránea que se refiere al ClienteID de la tabla cliente.
- **FOREIGN KEY**: Establecemos una relación con la tabla cliente, asegurando que cada orden esté asociada a un cliente existente.

- Procedimientos de Órdenes -Insertar Orden
  Definimos los procedimientos para insertar, actualizar y eliminar órdenes, de manera similar a los procedimientos para la tabla cliente.

```
--Insertar Orden
CREATE OR REPLACE FUNCTION insertarOrden(
    p_clienteID INT,
    p_fechaOrden DATE,
    p_monto DECIMAL(10,2)
)
RETURNS VOID AS
$$
BEGIN
    INSERT INTO ordenes (ClienteID, FechaOrden, Monto)
    VALUES (p_clienteID, p_fechaOrden, p_monto);
END;
$$ LANGUAGE plpgsql;
--Ejecutar el procedimiento de inserción de orden:
SELECT insertarOrden(1, '2025-01-06', 150.00);
```

Data Output | Messages | Notifications

Showing rows: 1 to 1 | Page No:

	insertarorden void
1	

- Procedimientos Actualizar Orden

```
--Actualizar Orden
CREATE OR REPLACE FUNCTION actualizarOrden(
    p_ordenID INT,
    p_nuevoMonto DECIMAL(10,2)
)
RETURNS VOID AS
$$
BEGIN
    UPDATE ordenes
    SET Monto = p_nuevoMonto
    WHERE OrdenID = p_ordenID;
END;
$$ LANGUAGE plpgsql;
```

Data Output | Messages | Notifications

```
CREATE FUNCTION

Query returned successfully in 121 msec.
```

- Procedimientos Eliminar Orden

```sql
--Eliminar Orden
CREATE OR REPLACE FUNCTION eliminarOrden(
    p_ordenID INT
)
RETURNS VOID AS
$$
BEGIN
    DELETE FROM ordenes WHERE OrdenID = p_ordenID;
END;
$$ LANGUAGE plpgsql;

--Ejecutar el procedimiento de eliminación de orden:
SELECT eliminarOrden(1);
```

Entrega Final

Instrucciones de Entrega:

1. **Objetivos**:

   Crear procedimientos almacenados para **insertar**, **actualizar**, **eliminar** y **consultar** registros en las tablas cliente y ordenes.

2. **Archivo de Script**:

   Los estudiantes deben escribir y guardar el código SQL con todos los procedimientos mencionados.

3. **Documento PDF**:

   Incluir las capturas de pantalla y explicaciones detalladas de los pasos realizados durante la tarea.

4. **Subida a GitHub**:

   Subir el script .sql y el documento PDF a un repositorio en GitHub para su REVISIÓN